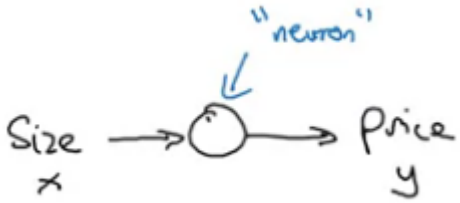
Week 1

What is a neural network?

- Neuron = takes an input and transforms it into output through a linear function



- Neural network (NN) = stacking neurons
    - one neuron for each recognized/classified feature
- Every input layer is connected to all neurons in a hidden layer

Supervised learning

- CNN = images
- RNN (recurrent) = temporal series (sequence data)
- Structured data = can be shown in a table
    - Every feature has a well-defined meaning)
- Unstructured data (audio, image, text)

Why is deep learning taking off?

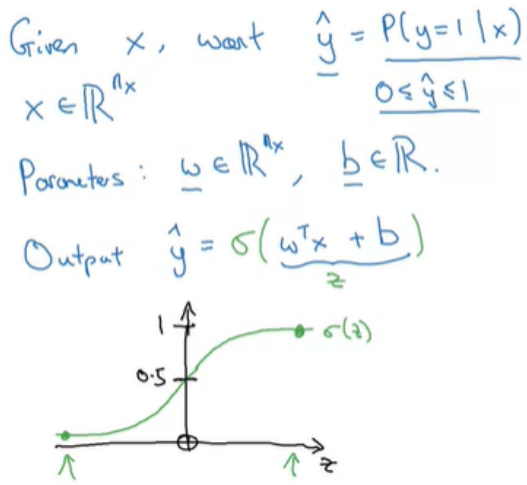
- m = size of the training set
- smaller training set = order of NN algorithms are not well-defined
    - we need good skills to determine features manually
- Bigger dataset = larger NN are better

Week 2

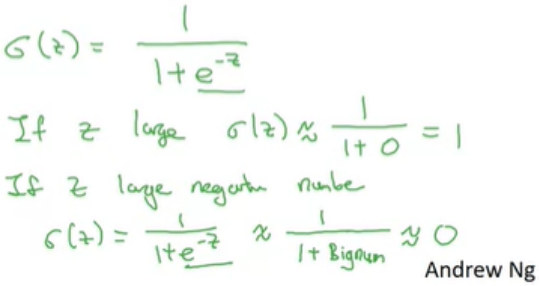
Binary Classification

- I want to iterate all datasets without a for loop
    - forward and backward propagation
- (x,y) pair > x is a value, and y is the label
- the dataset has 2 lines (x and y) and m columns

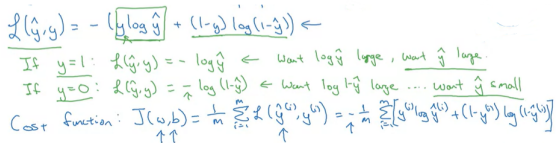
Logistic regression



- The sigmoid function goes from 0 to 1 = adequate to indicate a probability
- Objective = choose omega and b such that the sigma returns the probability of x return y as label

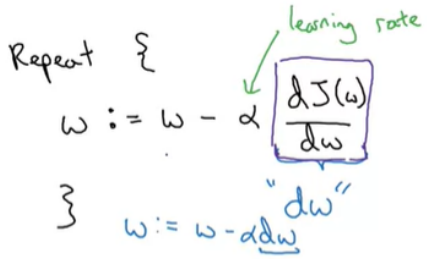


Logistic regression cost functions



Gradient descent

- dJ/dw = gradient

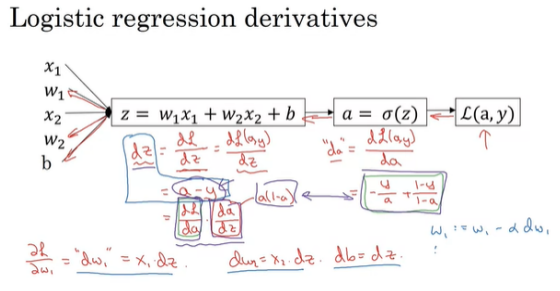


- J(w,b) = dependent variable

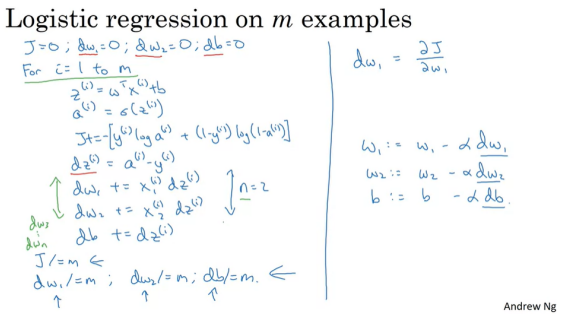
Computation graph

- J is the final step of calculation (cost function)
- We have w and b and compute J (forward propagation)
- We then have J and compute the derivatives in relation do w and b (backward propagation)

Logistic regression gradient descent



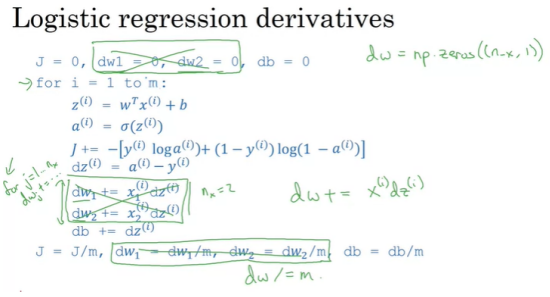
Gradient descent on m Examples



- there are 2 for loops (m examples and n dimensions of x)

Vectorization

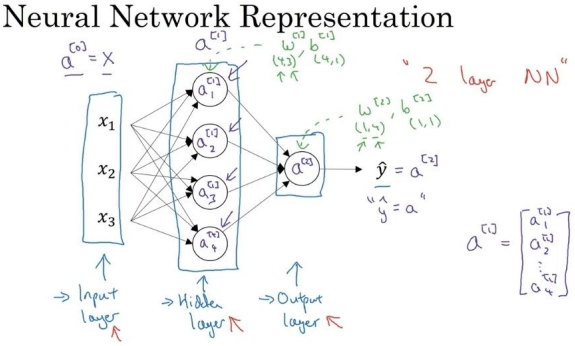
- saves time to compute z = w^Tx+b



Week 3

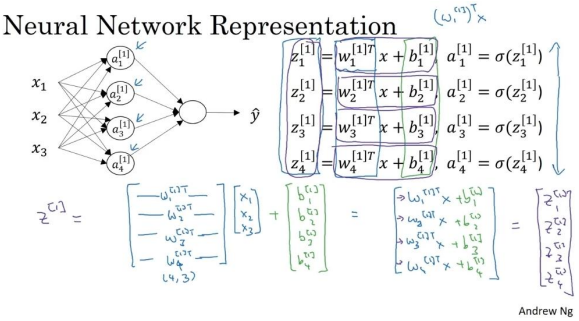
Neural Network Representation

- Superscript with brackets = layer
- Superscript with parenthesis = element in the dataset
- x = a[0] (activation parameters for the first layer)

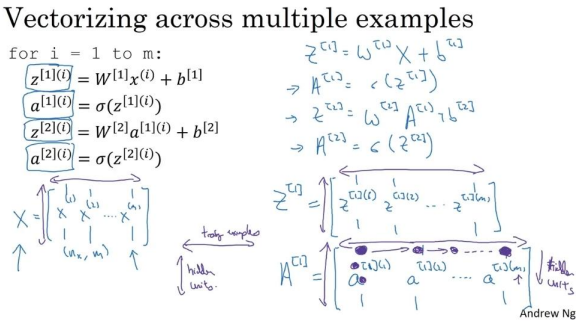


Computing a Neural Network's output

- Vectorizing logistic regression for multiple nodes (vertically stacked)
- Detail: this diagram shows for one input data x of 3 items each



Vectorizing across multiple examples



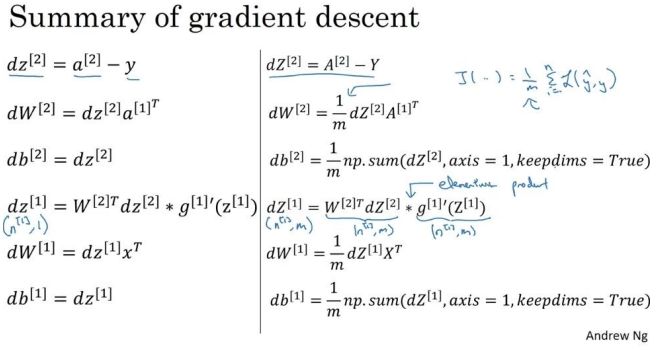
Activation functions

- Sigmoid function may not be appropriate >> good for binary classification at the output layer (returns 0 or 1)
- tanh goes from -1 to 1 >> good for hidden layers
    - The mean of the values is zero >> centring data (see 2nd course)
- if z is very large or very small, tanh and sigmoid is approximately zero >> slows down learning
    - use ReLu function >> dz = 1 for z positive or dz = 0 fr z negative
- Leaky Relu >> dz = -k, for k is small

Why use a non-linear activation function?

- If you use a linear function, the output of the NN is a linear function of the input X
    - So there's no difference in using one or more hidden layers
- But you can use it at the output layer if the predicted value is of the same nature as the input

Backpropagation with multiple layers and examples



Initiate with random values

- If all same units have the same function and the same initial weight, the result of the backdrop will be the same
- It's enough to initiate w non-zero and b thanks
- W must be small to avoid flat parts of sigmoid or tanh function >> could slow down learning

Week 4

Deep L-layer NN

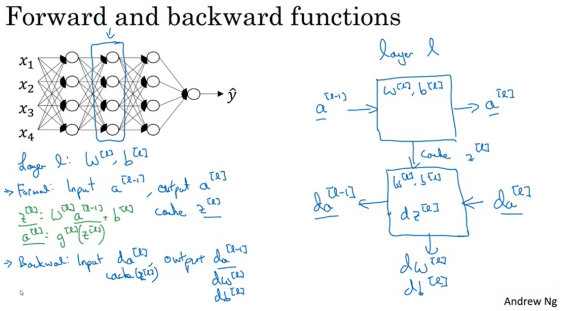
- L = number of layers
- n[L] = number of neurons in layer L
- First countable layer is not the input layer
    - input layer = 0
- a[L] = activation variable of layer L

Foward propagation in a deep NN

$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

- Different examples = different columns

Building blocks in deep NN



Hyperparameters

- Learning rate (alpha)
- How many iterations must I do
- How many hidden layers
- How many hidden units per layer
- What's the best activation functions